

LEGO® TC logo

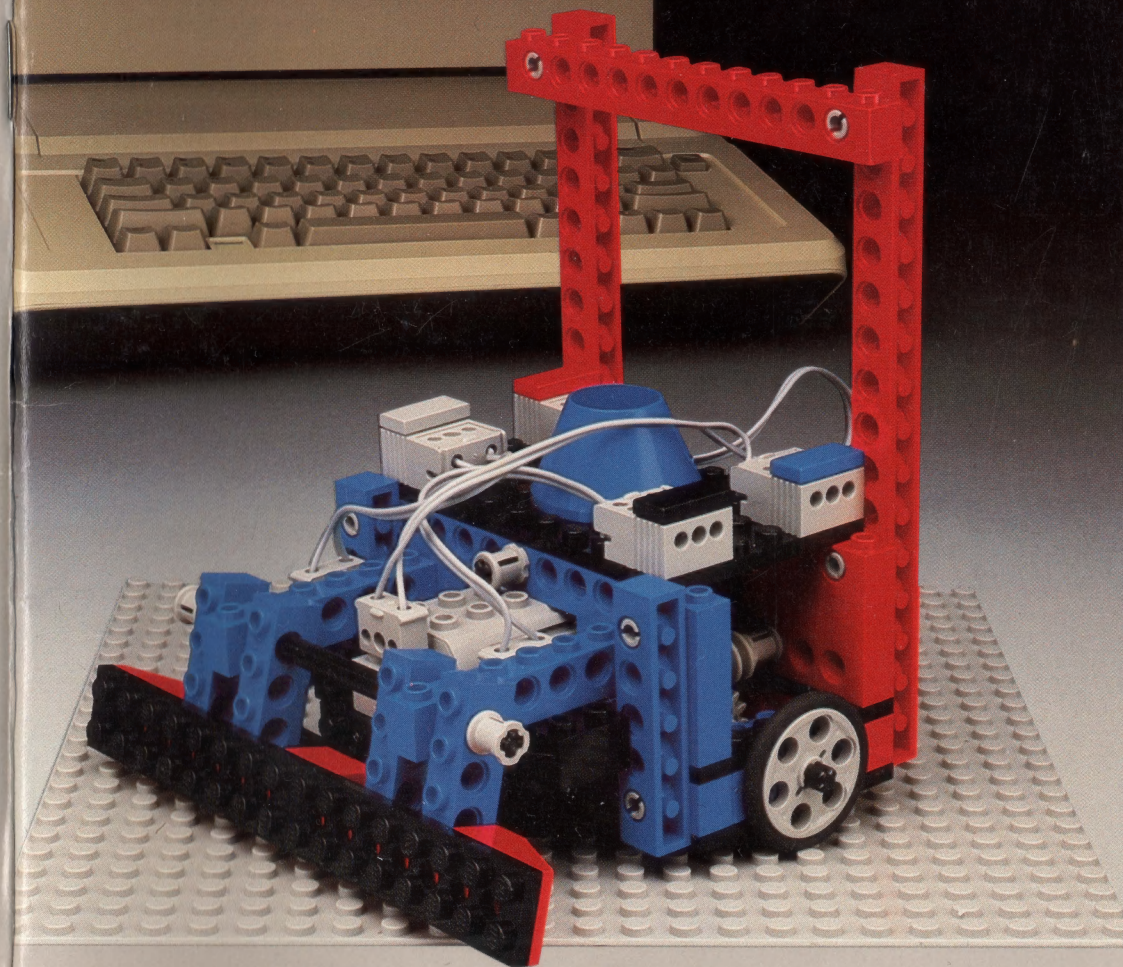
LEGO Dacta
Educational Products
555 Taylor Road
Enfield, CT 06082

LEGO Australia Pty. Ltd.
P.O. Box 639
Lane Cove
N.S.W. 2066



ISBN 0-914831-70-4
198325
Printed in U.S.A.

LEGO® TC logo



Teaching
the
Turtle



Teaching the Turtle

No part of the documents contained herein may be reproduced, stored in retrieval systems or transmitted in any form or by any means, photocopying, electronic, mechanical, recording or otherwise, without the prior approval in writing from the LEGO Group.

2nd revision, January 1989

ISBN 0-914831-70-4

LEGO® and the LEGO logo® are registered trademarks of INTERLEGO AG. ©1987, 1989 LEGO Group.

Apple and the Apple logo are registered trademarks of Apple Computer, Inc.

IBM is a registered trademark of International Business Machines Corporation.

MS-DOS is a registered trademark of Microsoft Corporation.

Meet the Turtle

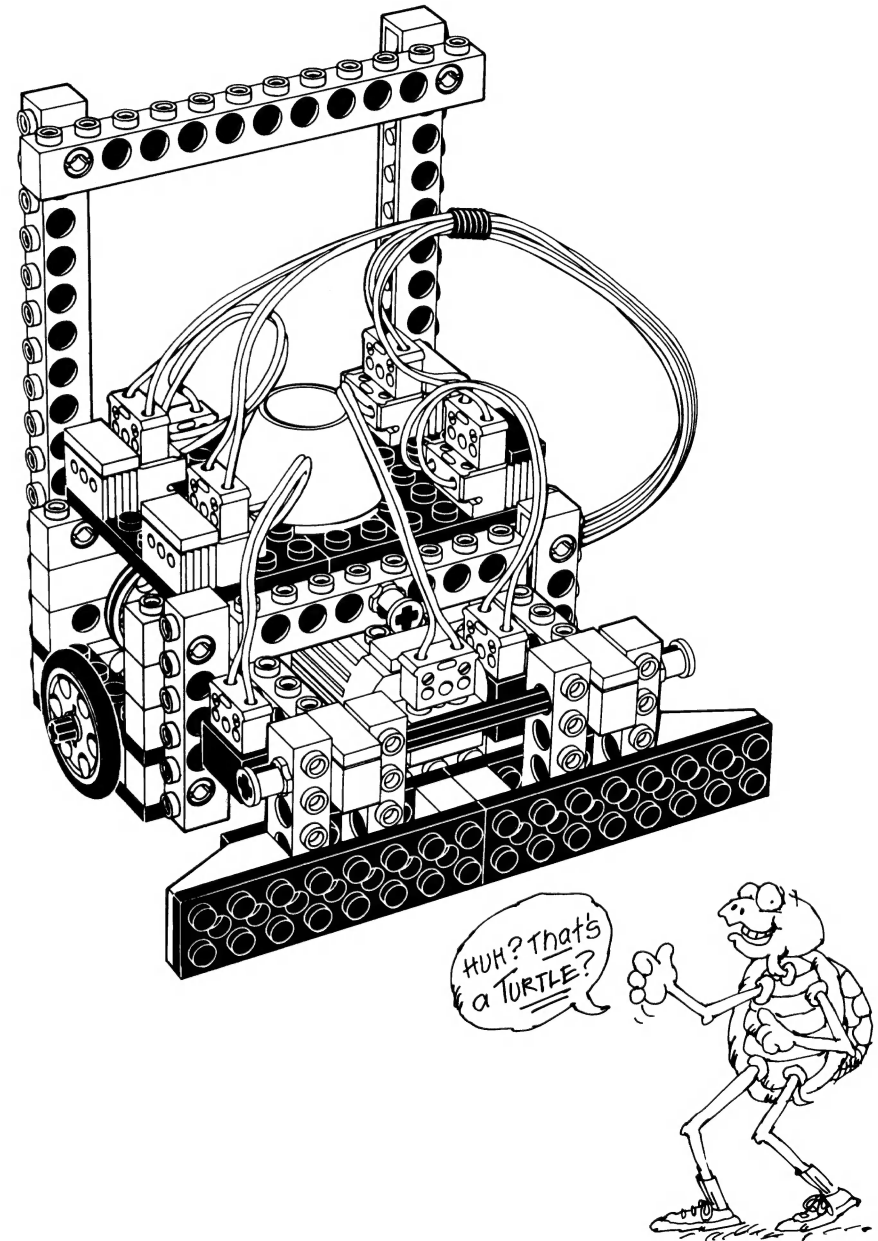
You are about to meet the LEGO® turtle!



Of course, the LEGO® turtle is not a real turtle, but it is like a real turtle in many ways. It can move around and “see” things and “touch” things.

By writing LEGO® TC logo programs, you can teach the LEGO® turtle how to behave. Your programs can make the turtle seem almost intelligent!

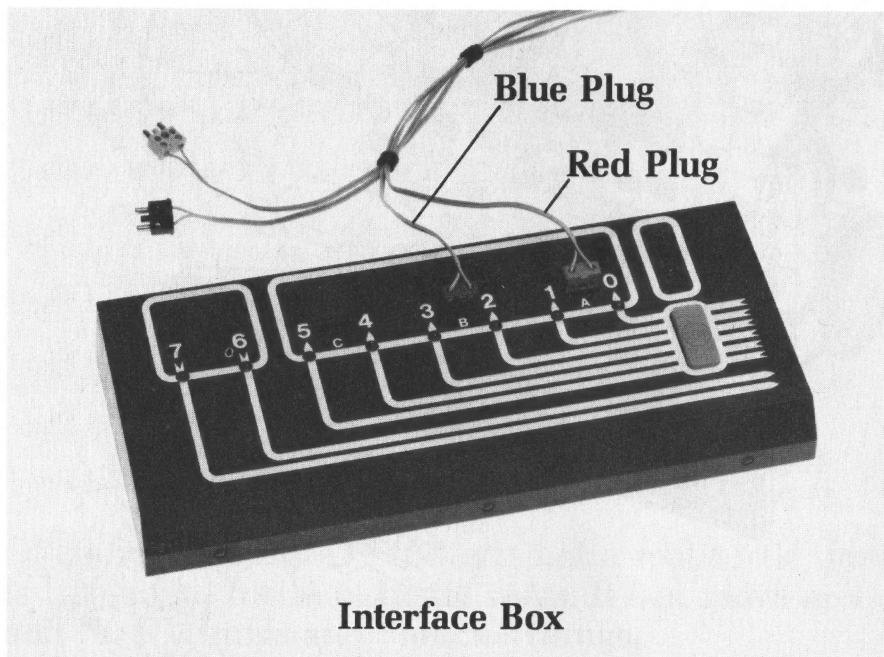
Activity Card 8 shows how to build the LEGO® turtle.



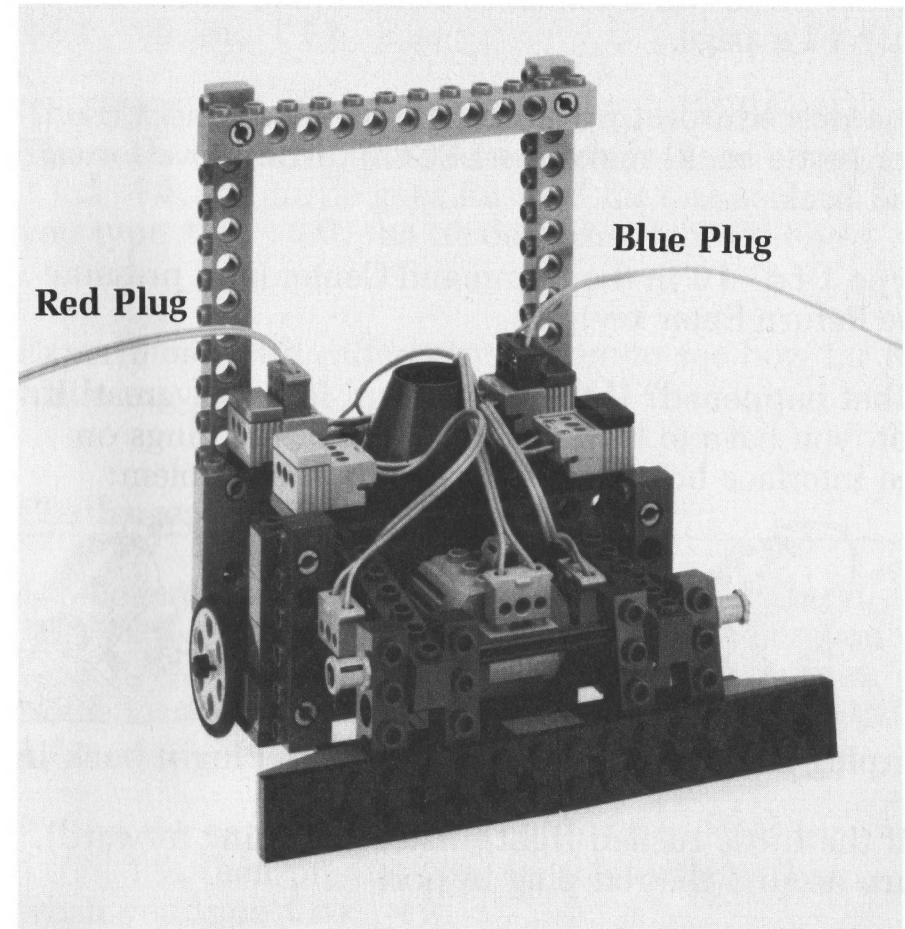
Connecting the Turtle

Before you can control the turtle from the computer, you must connect the turtle to the interface box. You should use the **turtle cable** to make the connection.

The turtle cable has four wires with plugs at each end. One end of the cable connects to the interface box. Connect the red plug to port A and the blue plug to port B. For now, leave the other two plugs unconnected.



Connect the other end of the cable to the outlets on top of the turtle. Connect the red plug to the outlet with the red plate. Connect the blue plug to the outlet with the blue plate.



The blue outlet connects to the front motor, which controls the left wheel. So the wire with the blue plug controls the left wheel. Similarly, the wire with the red plug controls the right wheel.

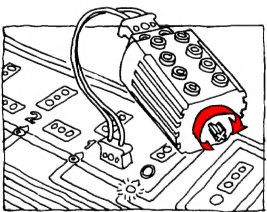
Turtle Commands

The **turtle** page on the LEGO® TC logo disk has special procedures that let you control the LEGO® turtle. Go to the **C o n t e n t s** menu and select the **t u r t l e** page.

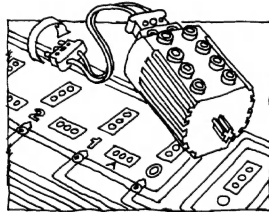
The procedures **t f d** (for **turtle forward**) and **t b k** (for **turtle back**) make the LEGO® turtle move forward and back.

Type **t f d 10** in the Command Center (and press the **R e t u r n** Enter key).

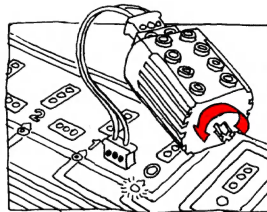
What happened? If the turtle went forward, great! If not, you have to turn around some of the plugs on the interface box. Here's how to fix the problem:



Unplug it



Flip it!



Plug it back in

*If the turtle turned right (instead of going forward), turn around the red plug in port A.

*If the turtle turned left, turn around the blue plug in port B.

*If the turtle went backwards, turn around the plugs in **both** port A and port B.

Try **t f d 10** again. Did it work this time?

Now try **t b k 15** to make the turtle move backward.

To make the LEGO® turtle turn, use the commands **t r t** (turtle right) and **t l t** (turtle left). Try **t r t 10** and **t l t 20**.

The LEGO® turtle commands are in units of time. There are 10 turtle units to a second. If you type **t f d 10**, the turtle goes forward for one second. If you type **t r t 10**, the turtle turns right for one second.

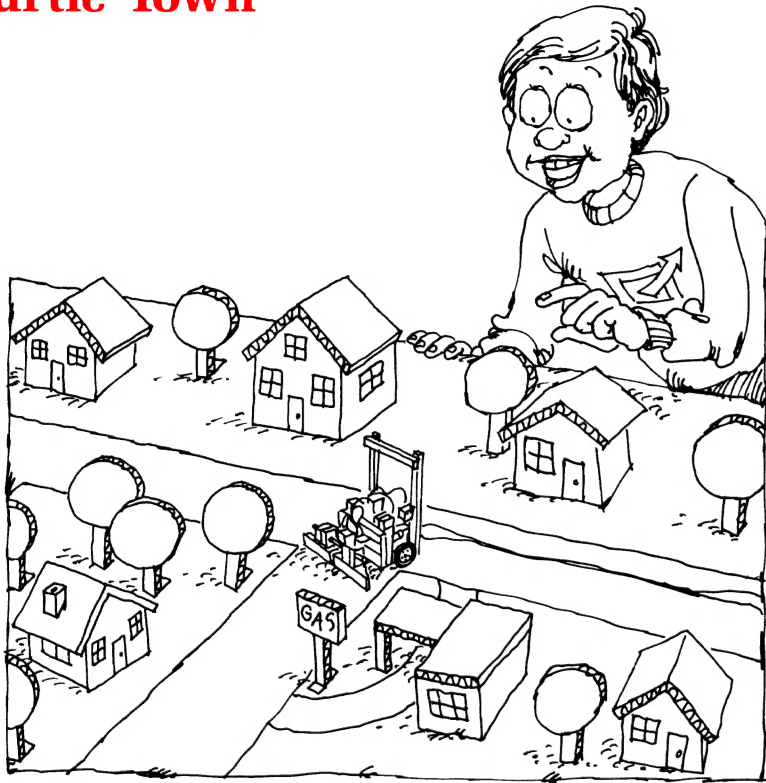
Experiment with different numbers to see how far the turtle moves!

Challenges

1. Measure how far the turtle goes when you type **t f d 10**. Then measure how far it goes when you type **t f d 20**. Record the results in your Inventor's Notebook. How far will the turtle go when you type **t f d 40**?

2. Run the same experiment with **t b k**. First try **t b k 10**, then predict how far the turtle will go when you type **t b k 40**.

Turtle Town



You can set up a “turtle town” in your classroom. Draw a plan for the town on a large piece of paper. Show the streets and places for houses, stores, and playgrounds. You can use LEGO® bricks or other materials (like milk cartons) to make buildings for your town.

Steer the turtle through the town by typing these commands in the Command Center: **tfd**, **tbk**, **trt**, and **tlt**. Make sure the streets are wide enough for the turtle!

You can write procedures that tell the turtle how to get to different places in the town. (See page 20 of the *Getting Started* book to learn more about procedures.)

Important note: There are already some procedures on the **turtle** page. You need these procedures to make the turtle work. Do not erase them.

When you add your own procedures, you should give a new name to the **turtle** page. That way, the original **turtle** page will not be changed (and you can use it again later).

Give the page a new name like **town**.
Type **namepage "town**
(or **np "town**) in the Command Center.

Now, you can write a procedure that moves the turtle from its home to the school. The procedure will depend on the layout of your turtle town. Here’s one possibility:

```
to home.to.school  
tfd 16  
trt 12  
tfd 27  
end
```


Drawing with the Turtle

You can draw shapes and pictures with the LEGO® turtle. Just place the turtle on top of a piece of paper, and put a soft-tipped marking pen through the hole in the turtle. Make sure the pen touches the paper.

Experiment with `tfd`, `tbk`, `trt`, `tlt`.

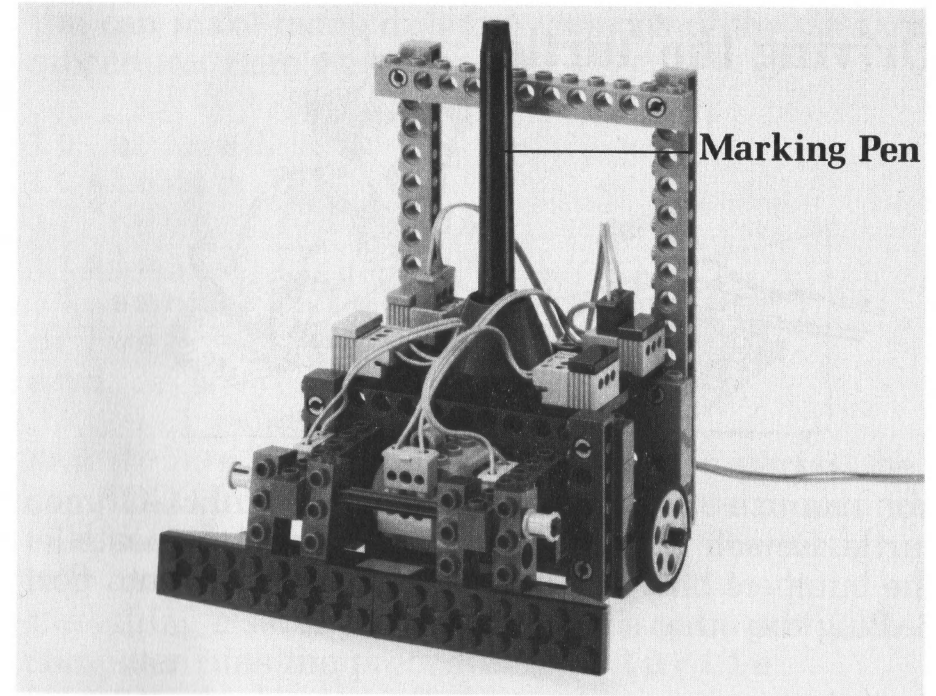
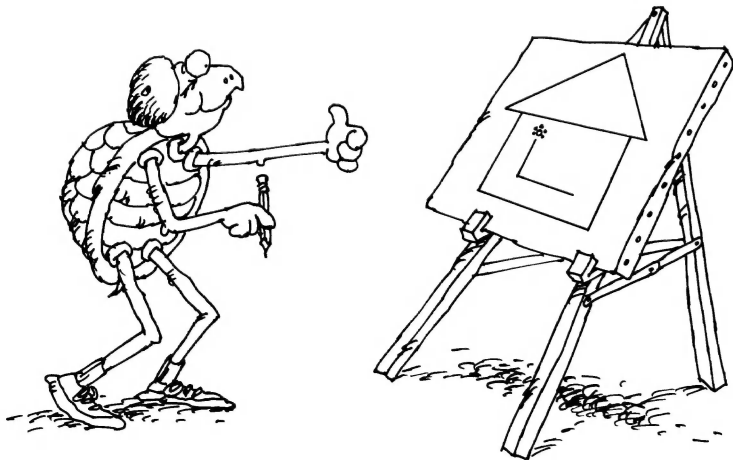
For example, try typing each of these in the Command Center.

```
tfd 10 trt 5 tfd 15
```

```
repeat 3 [tfd 10 trt 12]
```

```
repeat 4 [tfd 10 tlt 5]
```

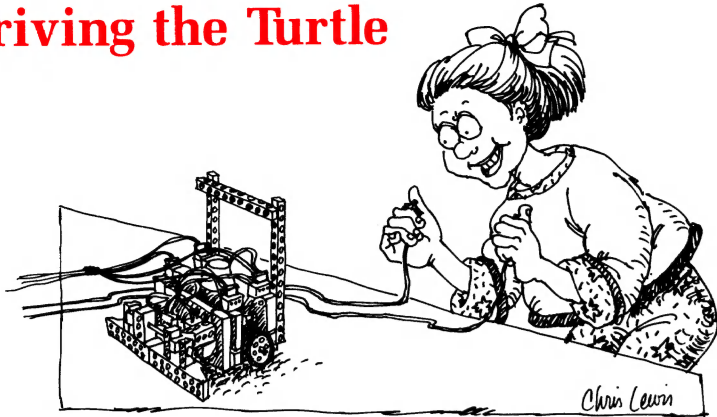
Remember that the inputs to `trt` and `tlt` are in units of time (ticks), not degrees. (See page 18 of *Getting Started* to learn more about `repeat`.)



Challenges

1. When you type `trt 10`, the turtle turns right for one second. How many degrees does it turn? What if you type `trt 5`? (Hint: Type `tfd 10 trt 10 tfd 10` to make the turtle draw an angle. Then use a **protractor** to measure the angle. Be careful: Make sure you measure the correct angle.)
2. What should you type to make the turtle turn 90 degrees (a **right angle**)?
3. Write a `square` procedure that makes the turtle draw a square.
4. Write a `circle` procedure.

Driving the Turtle



You can use touch sensors to “drive” the LEGO® turtle. Detach both touch sensors from underneath the bumpers on your turtle. Plug one sensor into port 6. Plug the other sensor into port 7.

```
to drive
  listento 6
  if sensor? [tfd 1]
  listento 7
  if sensor? [trt 1]
  drive
end
```

Try **drive**. When you press the sensor in port 6, the turtle moves forward a little bit. When you press the sensor in port 7, the turtle turns right a little bit.

What happens when you press both sensors?

Remember: To stop a recursive procedure, hold down the ⌘ Ctrl key and press the **S** key (**S** for **Stop**). If the motors are still going, type **all off**.

You can make many different versions of the **drive** procedure. Here’s another one:

```
to drive2
  listento 6
  if sensor? [gofd]
  listento 7
  if sensor? [offturtle]
  drive2
end
```

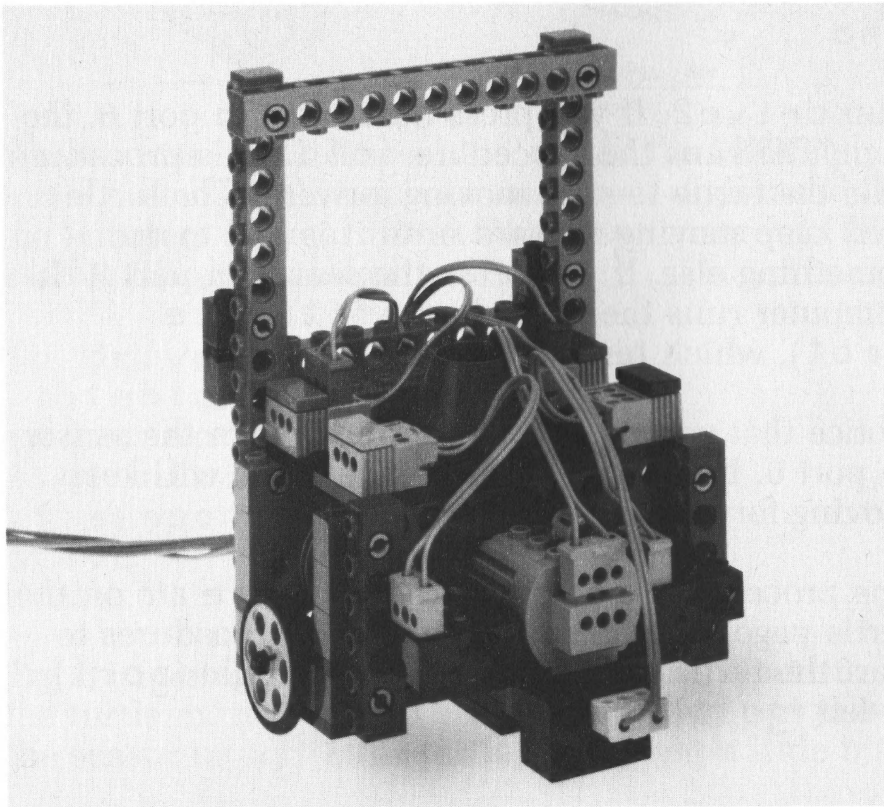
Run **drive2**. If you press the sensor in port 6, the computer runs the procedure **gofd**. This procedure tells the turtle to start moving forward. The turtle will keep moving forward until it is told to do something else. If you press the sensor in port 7, the computer runs the procedure **offturtle** (or **ot**), which turns off the turtle.

Notice that you do not have to hold down the sensor in port 6. If you press it once, the turtle will keep moving forward!

The procedures **gofd** and **offturtle** are on the turtle page. The turtle page also has procedures to start the turtle moving back (**gobk**), right (**gort**), or left (**gol t**). Try them!

Seeing-Eye Turtle

If you add an optosensor to the LEGO® turtle, the turtle will be able to “see”! Of course, the turtle’s “eyesight” is not very good. But it can tell the difference between black surfaces and white surfaces.



To start, connect the optosensor to the turtle, as shown on Activity Card 8. Then attach one end of the wire with the black plug into the black outlet on top of the turtle. Attach the other end into port 6.

Take a piece of white paper and a piece of black paper. Overlap the two so that most of each page is showing. Place the turtle on the white page.

Say that you want the turtle to move forward until it reaches the black paper. Here’s one way to do it:

```
to look.for.black  
  gofd  
  listento 6  
  waituntil [sensor?]  
  offturtle  
end
```

`look.for.black` starts the turtle moving forward, then it waits until the sensor “sees” the black paper. When the sensor sees black, `look.for.black` tells the turtle to stop (`offturtle`).

BUG BOX



If the turtle stops before the black line pick it up, put it down and try the procedure again. Picking up the turtle “reinitializes” the optosensor. See Chapter 2 of the *Reference Guide*.

Turtle in a Box

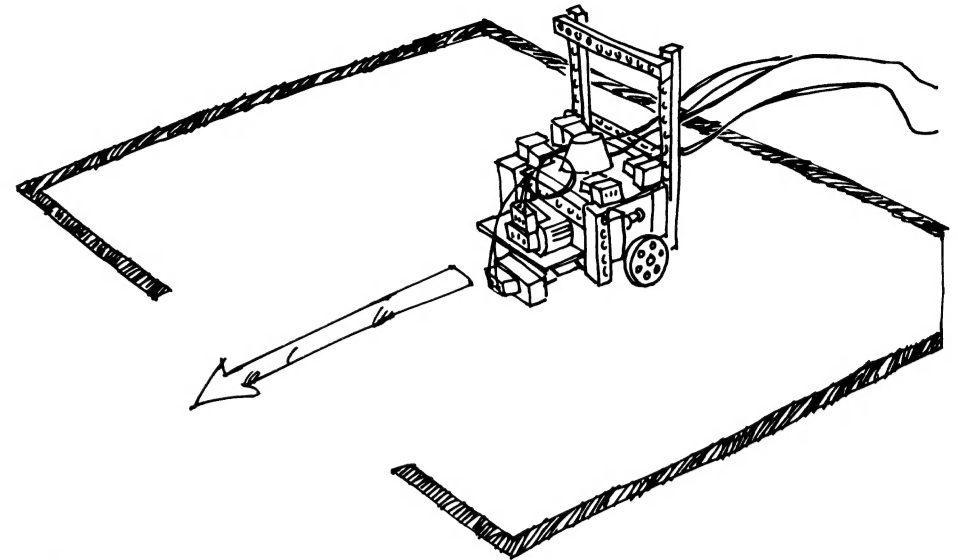
Imagine a turtle caught in a box. It wanders around looking for an opening. Whenever the turtle bumps into a wall, it turns around and looks in a new direction.

You can **simulate** this situation with the LEGO® turtle and optosensor. To make a “turtle box,” you can use black tape on a white surface (see diagram on next page). Then write a procedure like this:

<code>to wander</code>	
<code> go fd</code>	◆ Tell turtle to go forward
<code> listen to 6</code>	
<code> wait until [sensor?]</code>	◆ Wait until turtle sees a “wall”
<code> off turtle</code>	◆ Turn off the turtle
<code> tone 400 10</code>	◆ Sound a beep
<code> tlt 10</code>	◆ Tell the turtle to turn
<code> wander</code>	
<code>end</code>	

The **tone** primitive takes two inputs. The first input specifies the **pitch** of the tone and the second input specifies the **duration** of the tone (see the *Reference Guide*).

Now put a doorway in the turtle’s box so that the turtle can “escape.” How long does it take the turtle to escape? Try it several times, and record the results in your Inventor’s Notebook.



Challenges

1. Try changing the size of the doorway in the turtle box. Does the size of the door change the turtle’s “escape time”?
2. How can you change the **wander** procedure so that the turtle escapes more quickly? (The turtle must use the doorway!)
3. Here’s a tough challenge: Program the turtle so that it sounds several beeps and automatically stops when it gets out of the box. (Hint: Use **timer** to keep track of how long it has been since the turtle saw a black line. If it hasn’t seen a line in a long time, it must have escaped.)

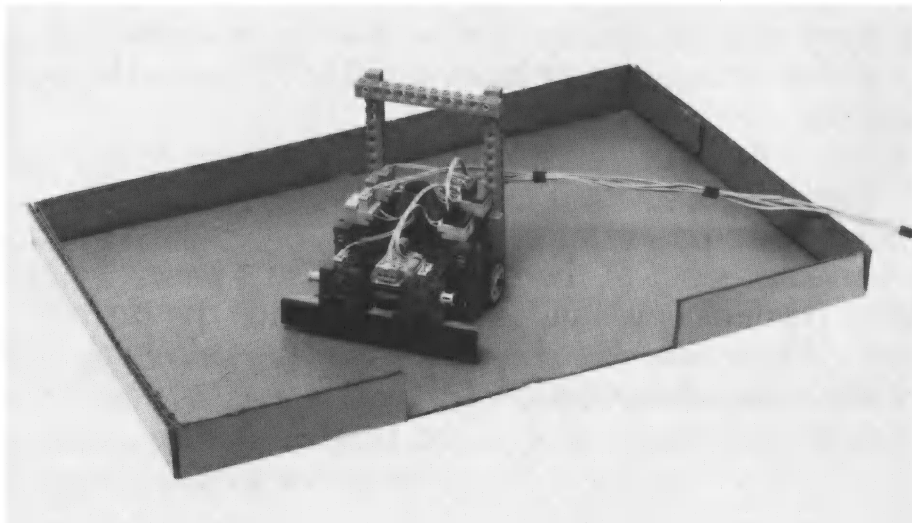
Turtle in a REAL Box

What if you put the LEGO® turtle inside a **real** box? You might use a cardboard box or a wooden crate. How can the turtle know when it bumps into a wall?

The optosensor will not be much help. The turtle needs **touch sensors**.

If you put a single touch sensor on the front of the turtle, you can use the same **wander** procedure as before (page 16). Try it! When the turtle bumps into a wall, it should turn and start “wandering” some more.

You can make the turtle “smarter” by giving it **two** touch sensors. Activity Card 8 shows how to make two “bumpers” for the turtle, with a touch sensor under each one.



You can program the “bumper turtle” to use both of its touch sensors. If the turtle “feels” a wall with its left bumper, it should turn to the right. If it “feels” a wall with its right bumper, it should turn left.

Attach one end of the wire with the grey plug into the grey outlet on top of the turtle. Attach the other end into port 7. Likewise, attach one end of the wire with the black plug into the black outlet on top of the turtle. Attach the other end into port 6. Then try **wander3**.

```
to wander3
  tfd 10
  listento 6
  if sensor? [trt 10]
  listento 7
  if sensor? [tlr 10]
  wander3
end
```

Challenge

Write a procedure that uses the touch sensors **and** the optosensor. You will need to attach the optosensor to the back of the turtle. For example, program the turtle to turn around when it bumps into a wall, and to sound a tone when it passes over a black line. (Hint: You can plug **both** touch sensors into the same port. The port will report **true** whenever **either** of the sensors is pressed.)

Follow the Line

How could you program the turtle to follow a line?

Here's one idea: Attach the optosensor to the front of the turtle and program the turtle to weave back and forth over the line. Say that the turtle begins by turning to the right. When it crosses the line, it should begin turning left. When it crosses the line again, it should begin turning right again, and so on.

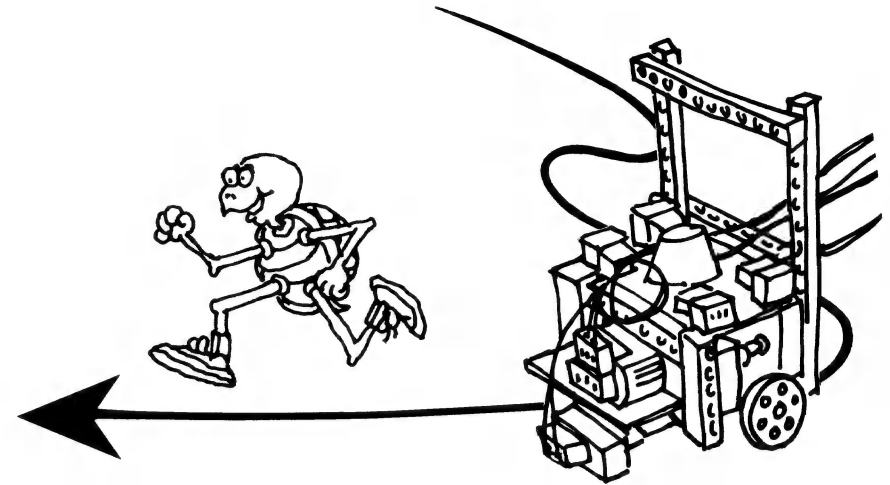
Your procedure could look like this:

```
to follow
talkto "A          ⚡ Talk to "red" motor
look.for.line
talkto "B          ⚡ Talk to "blue" motor
look.for.line
follow
end

to look.for.line
on
waituntil [floor = "black] ⚡ Until turtle
                                "sees" line
waituntil [floor = "white] ⚡ Until it
                                crosses the
                                line

off
end

to floor
listento 6
ifelse sensor? [output "black]
                [output "white]
end
```



Use black tape to make a line, then try the **follow** procedure.

Notice that the computer uses only one motor at a time. First it talks to the motor driving the left wheel, then to the motor driving the right wheel. This makes the turtle weave forward one way, then the other.

Using **gort** and **golt** would not work. The turtle would turn back and forth in the same spot.

Challenges

1. Change the **follow** procedure so that the turtle follows a **white** line on a **black** surface.
2. Make a piece of paper that is half black and half white. Program the turtle to follow the edge between the black and the white.

Turtle Procedures

It is no mystery how the “turtle commands” work. You can see the procedures on the `turtle` page!

There are ten procedures on the page. Let’s take a look at the `tfd` procedure:

```
to tfd :time
talkto [A B]
seteven
onfor :time
end
```

- ◆ Set both motors in the **even** direction
- ◆ Turn on motors for the indicated time

Since both motors are turning in the same direction, both wheels will turn in the same direction — and the turtle will move forward!

Type `tfd 20` and watch how the wheels turn. The number `20` replaces `:time` wherever `:time` appears inside the `tfd` procedure.

Of course, the direction of the motors depends partly on how you plugged in the turtle cable. That is why you had to adjust the plugs on the cable at the beginning of the turtle activities (page 6).

The `tbk` procedure is very similar to `tfd`, except both motors turn in the other direction (the **odd** direction). Try `tbk 20`.

`trt` is a bit trickier:

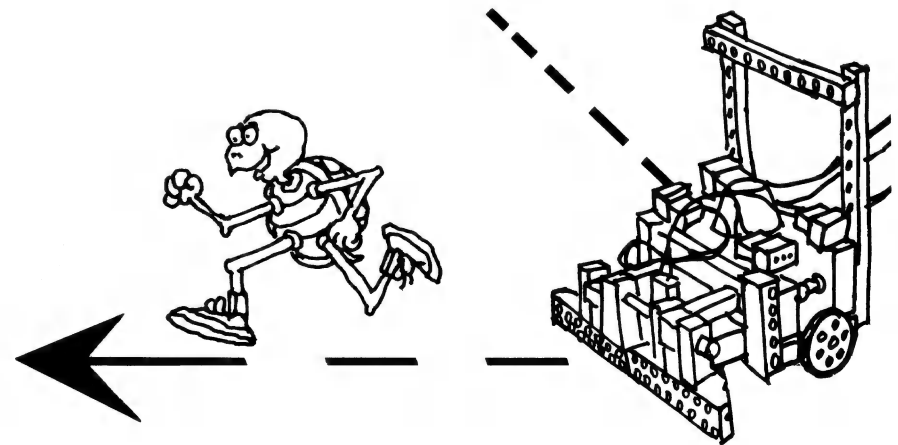
```
to trt :time
talkto "A
setodd
talkto "B
seteven
talkto [A B]
onfor :time
end
```

- ◆ Talk to the right motor (in port A)
- ◆ Set it in the backwards (odd) direction
- ◆ Talk to the left motor (in port B)
- ◆ Set it in the forward (even) direction
- ◆ Talk to both motors
- ◆ Turn them on for `:time`

The procedure sets each motor in a **different** direction. So one wheel will turn forward and the other wheel will turn backward. That makes the turtle turn in place.

`tlt` is similar to `trt` except that each motor (and each wheel) turns in the opposite direction.

Try `trt 20` and `tlt 20` and watch the wheels turn.

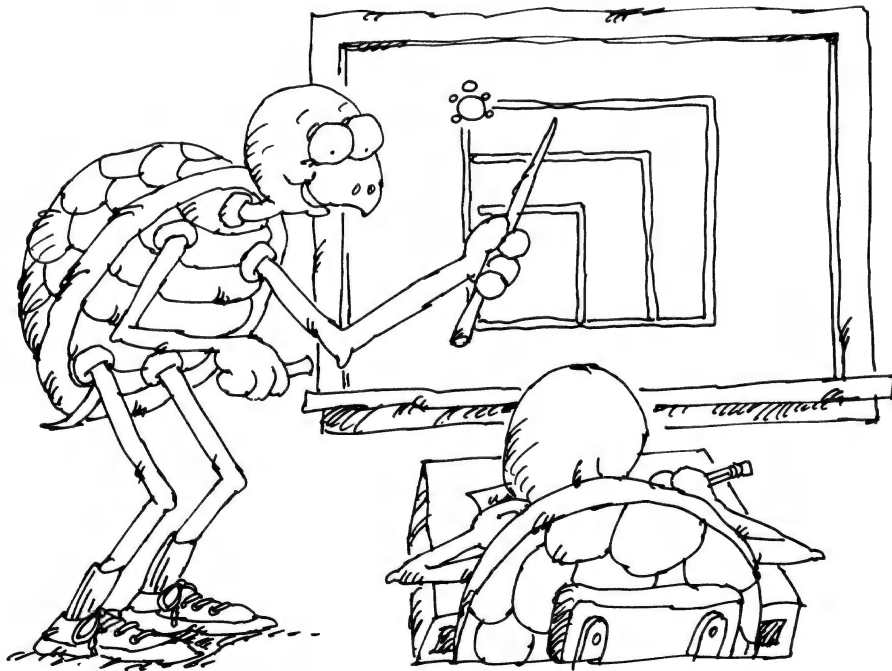


The next four procedures (`gofd`, `gobk`, `gort`, `golt`) are very similar to the first four. There is only one change: They all use `on` rather than `onfor`.

These procedures do not tell the turtle how long to move. They simply start the turtle moving. The turtle will continue moving in the same way until it is told to do something else.

The last two procedures (`offturtle` and `ot`) tell the turtle to stop. They simply turn off the turtle's two motors.

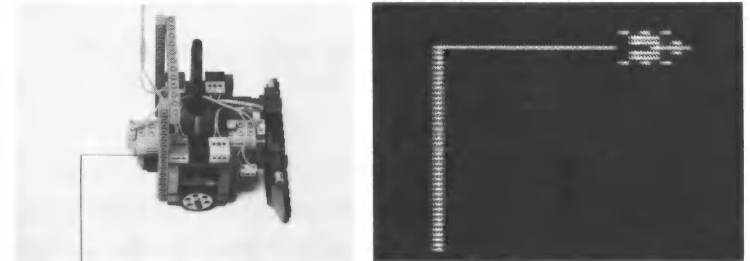
Try the procedures `gofd`, `gobk`, `gort`, `golt`, and `offturtle`.




Simulating the Turtle

Imagine that you had a very long turtle cable and you could send the LEGO® turtle into the next room. You would no longer be able to see the turtle. How could you know what the turtle was doing?

One way is to use the “screen turtle” (on the computer screen) to **simulate** the LEGO turtle. Whenever the **LEGO**® turtle goes forward, the **screen** turtle should go forward. Whenever the **LEGO**® turtle turns right, the **screen** turtle should turn right. And so on.




The computer screen is somewhat like a page of paper. The procedures are on one side. The screen turtle “lives” on the other side. To **flip** to the turtle side, hold down the  or Ctrl key and press the F key (F for Flip).

You can use the commands `fd`, `bk`, `lt`, and `rt` to move the screen turtle. For example, try these commands:

```
fd 50
rt 90
repeat 4 [fd 50 rt 90]
```

Type `cg` to clear the graphics.

To **flip** back to the procedure side, simply press -F Ctrl-F again.

To make the LEGO® turtle and the screen turtle move together, you will need to change some of the procedures on the **turtle** page. If you are not on the **turtle** page now, go back to the **Contents** menu and choose the **turtle** page.

Before making any changes give a new name to the **turtle** page.

```
namepage "turtle2
```

You can now make changes without affecting the original **turtle** page.

Let's start with the **tfd** procedure. To make the screen turtle move forward whenever the LEGO® turtle moves forward, all you have to do is add one line:

```
to tfd :time
talkto [A B]
seteven
onfor :time
fd :time      ◀ This is the new line
end
```

Whenever the LEGO® turtle moves forward for one second (**tfd 10**), the screen turtle will move forward 10 steps. Type **tfd 10** in the Command Center to see if the new procedure works.

It is easy to change the scale of the simulation.

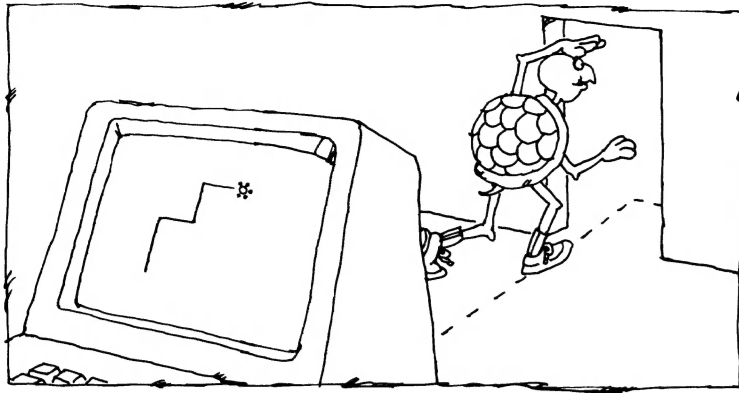
For example, change the new line to **fd :time * 2**. Now, whenever the LEGO® turtle moves forward for one second, the screen turtle will move forward 20 steps ($10 * 2 = 20$). So the screen turtle moves twice as far as before!

Changing the scale is particularly important for **trt** and **tlrt**. If you want your simulation to be “realistic,” then the screen turtle should turn 90 degrees whenever the LEGO® turtle makes a right-angle turn.

How can you do that? Start by typing **trt 10** in the Command Center. The LEGO® turtle should turn for ten ticks of the timer (one second). Let's say the turtle turns 80 degrees. That means the turtle turns eight degrees for every tick. ($80 \text{ degrees} / 10 \text{ ticks} = 8 \text{ degrees/tick}$).

For the simulation, the screen turtle should turn the same amount. You can change the **trt** procedure by adding one line:

```
to trt :time
talkto "A
setodd
talkto "B
seteven
talkto [A B]
onfor :time
rt :time * 8      ◀ This is the new line
end
```



Now, if you type `trt 10`, the input to `rt` will be $10 * 8 = 80$. So the screen turtle will turn 80 degrees, just like the LEGO® turtle!

Unfortunately, it is very difficult to make the simulation work for `gofd`, `gobk`, `golt`, and `gort`. These commands start the LEGO® turtle moving and leave it moving. It is difficult to simulate that with the screen turtle.

Challenges

1. Use the same approach to change `tbk` and `tlt`. When you are finished, the screen turtle should act just like the LEGO® turtle!
2. Write a `square` procedure so that the LEGO® turtle and the screen turtle both draw squares.
3. Put the LEGO® turtle back into your turtle town (page 8). Try the `home.to.school` procedure again. You can make a “blueprint” of your turtle town on the screen.

Changing Units

Right now, `tfd 10` moves the turtle forward for one second. Let's say you want to change the meaning of `tfd 10` so that it makes the turtle move forward for ten centimeters.

You can do this by changing the `tfd` procedure. Start by typing `tfd 10`. This turns on the turtle for ten ticks of the timer (one second). Let's say that the turtle moves forward six centimeters.

So the turtle takes $10/6$ or 1.66 ticks to move one centimeter. For the new version of `tfd` you want the turtle to move one centimeter when you type `tfd 1`. To make the turtle move one centimeter, you must turn on the motors for 1.66 “ticks.” So you must change the `onfor` line in the `tfd` procedure to:

```
onfor :time * 1.66
```

Of course, your numbers may vary.

Since the input is now in centimeters, you could change `:time` to `:centimeters` everywhere it appears in the definition for `tfd`. This change makes the procedure more readable. But you do not have to make this change. The procedure will work exactly the same.

Challenge

Change `trt` and `tlt` so that their inputs are in degrees not in units of time.

Glossary List

*The page numbers in the definitions refer to the **Getting Started** book pages where the words are first described.*

axle: A thin rod that connects to the middle of a wheel (or gear). The wheel and axle rotate together.

Command Center: The place where you type commands for the computer to carry out. The Command Center is at the bottom of the computer screen (page 11).

cursor: A small blinking light that indicates the current position on the computer screen (page 11).

debugging: The act of finding and fixing things that aren't working right. There might be "bugs" in either your LEGO® machines or your Logo procedures (page 22).

force: A push or pull that causes an object to speed up or slow down. Gravity and friction are examples of forces (page 6).

friction: A force that causes an object to slow down. Rough surfaces cause more friction than smooth surfaces.

gear: A wheel with "teeth." You can use gears to change the position, strength, speed, and direction of a rotating motion (pages 6, 40-43).

gear train: A combination of gears used for changing the strength and speed of a rotating motion. When a small gear "meshes" with a large gear, the large gear rotates more slowly, but with greater strength (page 40).

gravity: A universal force that pulls all objects toward one another. The Earth's gravity pulls all objects down toward the center of the Earth.

infrared: A type of light beyond the red end of the rainbow. The LEGO® optosensor sends out and detects infrared light (page 45).

interface box: The box that connects LEGO® motors, lights, and sensors to the computer. The interface box has 12 ports where you can plug in motors, lights, and sensors (page 9).

list: A special type of Logo object that combines several pieces of information into a single unit. Lists are enclosed in square brackets, like `[1 2 3]`. By using a list, you can manipulate many pieces of information at once, rather than separately (page 17).

motor: A machine that provides a rotating motion. LEGO® motors convert electrical energy to mechanical energy (page 6).

optosensor: A sensor that detects changes in light. The LEGO optosensor usually acts like a "dark sensor." It reports `true` when it "sees" something dark (page 28).

port: A set of holes in the interface box where you can plug a LEGO® wire. Some ports (**outlets**) send messages to motors and lights. Other ports (**inlets**) receive messages from sensors (page 9).

primitive: A procedure that is already built into the Logo language (page 21).

procedure: A named set of Logo instructions. When you type the name of a procedure in the Command Center, the computer runs the procedure (page 20).

Procedure Center: The place where you define procedures. The Procedure Center is at the top of the computer screen (page 21).

recursive procedure: A procedure that includes itself as part of its definition. Some recursive procedures run through the same “loop” over and over (page 26).

sensor: A device that sends information about the world to the computer. LEGO® TC logo has touch sensors and optosensors (page 8).

software: Instructions that tell the computer what to do. Software is often stored on a **disk**.

subprocedure: A procedure that is used within the definition of another procedure (page 24).

theory: A set of beliefs or guesses about the way things work (page 5).

LEGO® TC logo

LEGO Dacta Educational Products
555 Taylor Road
Enfield, CT 06082

LEGO Australia Pty. Ltd.
P. O. Box 639
Lane Cove
N.S.W. 2066

ISBN 0-914831-70-4
198325

Printed in U.S.A.